САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

КАФЕДРА ТЕОРИИ СИСТЕМ УПРАВЛЕНИЯ ЭЛЕКТРОФИЗИЧЕСКОЙ АППАРАТУРОЙ

Прус Анна Ивановна

Магистерская диссертация

Разработка нейронной сети прямого распространения для решения задач прогнозирования и аппроксимации

Направление 010900

Прикладная математика и физика

Математические и информационные технологии

Козынченко В. А.

Научный руководитель,
кандидат физ.-мат. наук,
доцент

Санкт-Петербург

2016

Abstract

Master's thesis is devoted to application of neural networks in problems of prediction and approximation. A batch method of training a network of linear approximation is offered and computational algorithm training is developed. Multilayer perceptron and modification of the network of the linear approximation are realized in the programming environment Matlab. Realization of computing experiment confirms the effectiveness of neural networks considered in the thesis.

Аннотация

Магистерская диссертация посвящена применению нейронных сетей в задачах прогнозирования и аппроксимации. Предложен пакетный метод обучения сети линейной аппроксимации и разработан вычислительный алгоритм обучения. В среде программирования Matlab реализован многослойный персептрон и модификация сети линейной аппроксимации. Проведенный вычислительный эксперимент подтверждает эффективность рассмотренных в диссертации нейронных сетей.

Содержание

Введение

С момента создания первых искусственных нейронных сетей они используются как для решения различных прикладных задач, так и для изучения возможных правил функционирования мозга. Нейронные сети успешно применяются в широком спектре приложений, таких как распознавание образов, прогнозирование, сжатие данных, задачи управления и другие. Одним из важных направлений практического использования искусственных нейронных сетей являются задачи прогнозирования. к задачам прогнозирования относятся задачи прогнозирования курса валют, прогнозирование курса ценных бумаг на фондовом рынке, прогнозирование цен на сырье и другие товары, задачи выявления страхового мошенничества и мошенничества при банкротствах, задачи определения свойств новых материалов по их структуре, прогнозирование нагрузок энергетических систем, предсказание платежеспособности кредиторов и состоятельности "start-up" проектов. Обобщая области применения искусственных нейронных сетей, можно утверждать, что искусственные нейронные сети могут применяться при решении любых задач для решения которых отсутствует адекватная математическая модель, в том числе недостаточно данных для эффективного применения статистических методов их анализа. Таким образом, применение искусственных нейронных сетей в задачах прогнозирования является актуальным.

Формально, задача обучения нейронной сети в задачах прогнозирования, формулируется как задача аппроксимации. Необходимо построить нейронную сеть (аппроксимирующую функцию), которая будет принимать совпадающие значения (с заданной точностью) не только на данных участвовавших в обучении (задача приближенной интерполяции), но и на данных контрольного множества не участвовавших в обучении. Задача распознавания образов также может быть формально поставлена как задача

аппроксимации. В задачах аппроксимации используется обучение с учителем, то есть для каждого обучающего входного вектора имеется обучающий выходной вектор.

Для решения задач прогнозирования (аппроксимации) используются нейронные сети прямого распространения сигнала такие как многослойный персептрон, сети радиальных базисных функций, машины опорных векторов. Наиболее часто в задачах аппроксимации используется многослойный персептрон. При обучении многослойного персептрона, как правило, используется хорошо зарекомендовавший себя на практике метод обратного распространения ошибки. Этот метод используется для численного определения градиента целевой функции (суммы квадратов ошибок), что позволяет использовать для обучения сети направленные методы спуска, в том числе метод наискорейшего спуска. Использование такого подхода к обучению многослойного персептрона на практике показало высокую эффективность, однако имеются и существенные недостатки. Прежде всего, это возможность паралича обучения при попадании в локальный минимум целевой функции. Выход из локального минимума без существенного ухудшения значений целевой функции, что фактически означает обучение заново, в некоторых случаях затруднителен. Второй проблемой обучения многослойного персептрона с использованием любых направленных методов минимизации суммы квадратов ошибок является зависимость архитектуры сети от начальных данных. При малом количестве нейронов скрытого слоя персептрон невозможно обучить на данных обучающего множества, причем причина неудачи обучения не всегда ясна. В ряде случаев такой причиной может быть малое время обучения. При избыточном числе нейронов скрытого слоя возможно избыточное обучение, при котором сеть успешно обучается на данных обучающего множества, но показывает плохие результаты на данных контрольного множества. В этих случаях необходимо увеличение или уменьшение числа нейронов скрытого слоя, а затем обучение сети заново, что приводит к дополнительным временным затратам на

обучение сети. Эти проблемы обучения многослойного персептрона делают актуальным разработку новых нейронных сетей прямого распространения обучающихся с учителем.

Наряду с многослойным персептроном в магистерской диссертации рассматривается двухслойная сеть линейной аппроксимации, которая обеспечивает разбиение входных обучающих данных n-мерными симплексами и их аппроксимацию гиперплоскостями на каждом симплексе. Обучение сети линейной аппроксимации происходит одновременно с формированием её структуры, когда в процессе обучения в сеть добавляются новые нейроны и изменяются веса связей остальных нейронов. Такой подход позволяет полностью решить проблему зависимости архитектуры сети от обучающих данных. Предложенный ранее последовательный метод обучения сети линейной аппроксимации показал хорошие результаты в сравнении с многослойным персептроном. Значительным недостатком сети линейной аппроксимации является существенно большее количество нейронов по сравнению с многослойным персептроном. Для сети линейной аппроксимации актуальна разработка более эффективных методов обучения, обеспечивающих функционирование сети с меньшим количеством нейронов. В настоящей работе предлагается пакетный метод обучения сети линейной аппроксимации, который показал лучшие результаты по сравнению с последовательным методом обучения.

Постановка задачи

Целью магистерской диссертации является совершенствование алгоритмов обучения нейронных сетей для решения задач аппроксимации.

В работе ставятся следующие задачи:

1) формулирование основных проблем обучения многослойного персептрона и создание компьютерной модели многослойного персептрона, в которой реализованы имеющиеся методы;

2) разработка улучшенного метода обучения сети линейной аппроксимации, который обеспечит обучение нейронной сети с меньшим количеством нейронов;

3) реализация улучшенного метода обучения сети линейной аппроксимации;

4) проведение вычислительного эксперимента.

Обзор литературы

Вопросы применения искусственных нейронных сетей описываются а большом числе работ. Достаточно представительный перечень литературы указан в библиографии работы [1]. Фундаментальный труд С. Хайкина является в настоящее время наиболее подробным систематическим изложением теории искусственных нейронных сетей. Теория многослойного персептрона, вопросы его обучения и практического применения изложены в работах [1]-[10], [12]-[14], [16]-[17]. Проблемы, возникающие при обучении многослойного персептрона и пути их решения, методы изменения архитектуры сети в связи с невозможностью обучения на данных обучающего множества и избыточным обучением наиболее подробно рассматриваются в [2], [6], [10]. Архитектура и вопросы обучения сети линейной аппроксимации рассмотрены в [18].

Глава 1. Многослойный персептрон

1.1. Искусственный нейрон

На рисунке 1 показана структура искусственного нейрона с n входами. Каждый входной канал i может передавать вещественное значение x_i. Активационная функция f вычисляется в теле нейрона и может быть выбрана произвольно. Как правило входные сигналы имеют соответствующий вес, что означает, что компоненты вектора входного сигнала x_i умножается на соответствующий вес w_i. Переданная информация обычно объединяется в нейроне с помощью взвешенного суммирования и затем вычисляется активационная функция.

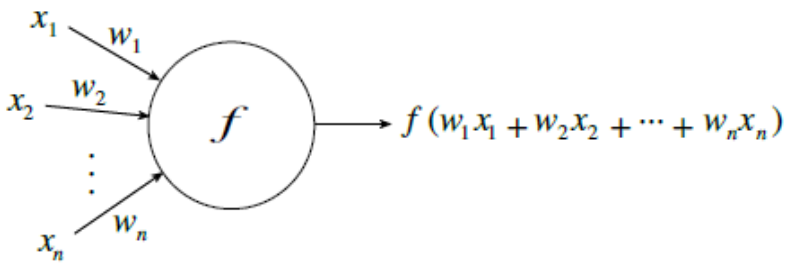

Рис.1. Модель абстрактного нейрона.

Если мы воспринимаем каждый нейрон в нейронной сети как элементарную функцию со способностью трансформировать свой входной сигнал в точно определенный выходной сигнал, то нейронная сеть является композицией активационных функций с набором параметров, которыми являются весовые коэффициенты связей. Различные модели искусственных нейронных сетей различаются в основном активационными функциями нейронов сети, способом межнейронных связей и по времени передачи информации.

В большинстве случаев предполагается, что каждый нейрон предоставляет аддитивный вклад для выходного нейрона, с которым он

соединен. Полный выходной сигнал k-го нейрона является значением функции активации от взвешенной суммы отдельных выходов от каждого подсоединенного нейрона плюс смещение или сдвиг:

$$s_k(t) = f(\sum_i w_{jk}(t) y_j(t) + \theta_k(t)), \quad (1)$$

Вклад положительных весов w_{jk} будет считаться возбуждением, а вклад отрицательных весов w_{jk} торможением. В некоторых случаях используются более сложные правила для объединения входящих сигналов, которые делятся между собой на возбуждающие и тормозящие входные сигналы.

Часто функция активации является неубывающей функцией от всего входящего сигнала нейрона:

$$y_k(t+1) = \mathcal{F}_k\big(s_k(t)\big) = \mathcal{F}_k\left(\sum_j w_{jk}(t) y_j(t) + \theta_k(t)\right), \quad (2)$$

В качестве функций активации (рис.2) часто используются следующие типы функций: жестко ограниченная пороговая функция (*sign* - функция), или линейная, или полулинейная или плавно-ограниченная пороговая функция. Для плавно-ограниченных функций часто *sigma*- функция выглядит так:

$$y_k = \mathcal{F}(s_k) = \frac{1}{1 + e^{-s_k}}, \quad (3)$$

В некоторых приложениях используется гиперболический тангенс, получающий выходные значения в диапазоне *[-1, +1]*.

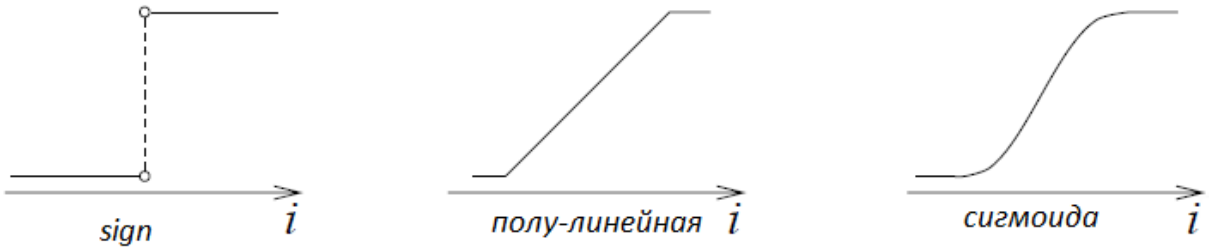

Рис.2 Графики некоторых функций активации.

В некоторых случаях выходной сигнал может быть стохастической функцией входного сигнала. В таком случае активация нейрона не определяется детерминировано - определенными входными нейронами, но входные нейроны определяют возможность p, того, что нейрон получит высокое значение активации:

$$p(y_k \leftarrow 1) = \frac{1}{1+e^{-s_k/T}}, \tag{4}$$

где T (температура) является параметром, который определяет наклон вероятностной функции.

1.2. Архитектура многослойного персептрона

В 1958 году Розенблатт, американский психолог, предложил персептрон, как более общую вычислительную модель, чем модель Маккалока-Питтса. Существенный новшеством стало введение численных весов и специальной конфигурации соединительных связей. В оригинальной модели Розенблатта вычислительные нейроны - это пороговые и соединительные элементы, определенные стахостически. Обучение происходит за счет адаптации сети и настройки весов с помощью числового алгоритма. Модель Розенблатта была доработана и усовершенствована в 1960 годы, и ее вычислительные свойства были тщательно

проанализированы Минским и Папертом. В дальнейшем модель Розенблатта стала называться классическим персептроном.

Классический персептрон сложнее, чем однослойная сеть с пороговой функцией активации. В своей простейшей форме она состоит из *N* элементов входного слоя, которые подаются на *K* элементов скрытого слоя, а затем, на выходной одиночный нейрон. Целью работы персептрона является изучение заданного преобразования $f: \{-1; 1\}^N \to \{-1; 1\}$ с использование обучающих примеров с входящими сигналами *x* и соответствующими выходящими сигналами *y=f(x)*. В первоначальном определении, активационной может быть любая функция , но процедура обучения регулирует только веса связей с выходным нейроном. В зависимости от функции активации персептроны могут быть сгруппированы в различные семейства. В 1969 году Минский и Пэйперт описали количество таких семейств и их свойства.

В настоящее время под многослойным персептроном обычно понимают сеть прямого распространения сигналов, без обратных связей, состоящую из нескольких слоев нейронов с сигмоидальной функцией активации. Каждый нейрон имеет связи со всеми нейронами предыдущего слоя и со всеми нейронами следующего слоя. Нейроны принадлежащие одному слою не имеют связей между собой. Все слои многослойного персептрона кроме последнего называются скрытыми.

Таким образом, многослойный персептрон имеет слоистую структуру (рис.3). Каждый слой состоит из нейронов, которые получают свои входные сигналы от стоя нейронов, расположенного ниже, и посылают свои выходные значения на слой, который соответственно находится выше. Нет никаких соединительных связей в пределах одного слоя. Входные сигналы подаются сразу на первый слой со скрытыми нейронами, т.к. самый первый входной слой служить лишь для инициализации входного вектора, никакой обработки сигнала в этом слое не происходит. Выходы скрытых нейронов в свою очередь распределяются по следующему свою скрытых нейронов, и так

происходит до тех пор, пока не достигается последний слой скрытых нейронов.

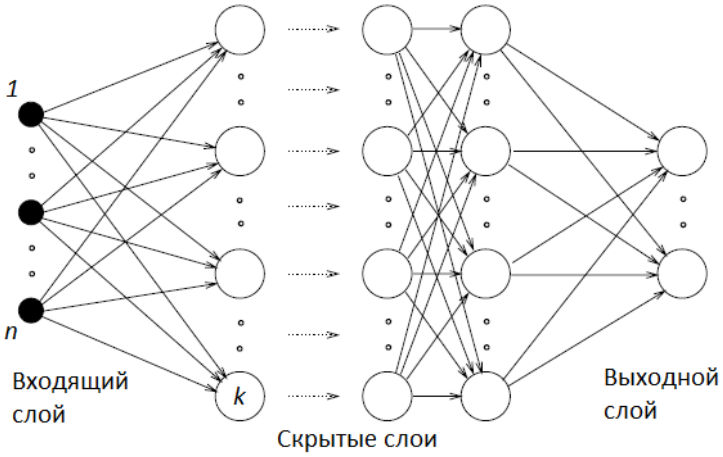

Рис. 3. Обобщенный вид многослойного персептрона.

1.3. Геометрическая интерпретация

Простой персептрон представляет собой искусственный нейрон с пороговой функции активации и n вещественными входными сигналами (x_1, x_2,..., x_n). Выходной сигнал равен единице, если выполняется условие $\sum_{i=1}^{n} w_i x_i \geq \theta$, а в противном случае выходной сигнал равен нулю.

Способ получения входных сигналов не имеет значение, приходят ли они от других персептронов или от другого слоя вычислительных нейронов. Геометрическая интерпретация обработки, выполняемая с помощью персептронов такая же, как и у нейронов Маккалока-Питса. Персептрон разделяет входное пространство на два полупространства. Для точек, принадлежащих одному полупространству выходной сигнал нейрона равен нулю, а для точек принадлежащих второму полупространству выходной сигнал равен единице.

Рисунок 4 показывает это для случая двух переменных x_1 и x_2. Показан результат работы простого персептрона с единичным порогом, у которого соприкасаются два ребра с весами *2* и *0.9*.

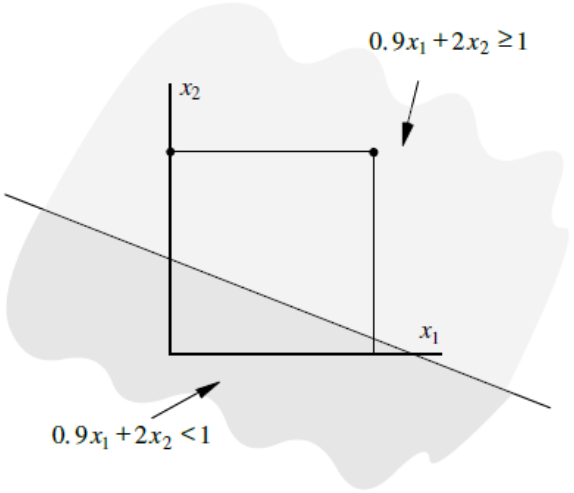

Рис. 4. Разделение входного пространства простым персептроном.

Можно выбирать произвольное разделение пространства входных сигналов, регулируя весовые коэффициенты нейрона.

Во многих случаях удобнее всего иметь дело с персептроном, у которого нулевая пороговая функция. Это соответствует линейному разделению, которое вынуждено проходить через начало входного пространства. Два персептрона на рисунке 5 эквивалентны. Порог персептрона слева был преобразован в вес дополнительного входного канала, соединенного с константой равной единице. Этот дополнительный вес, подсоединенный к константе называется смещением элемента.

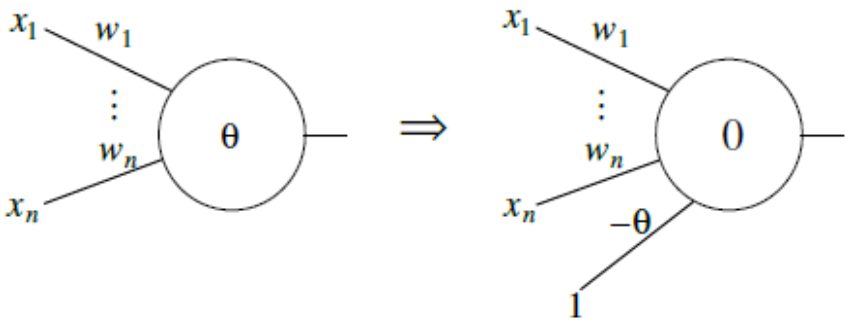

Рис. 5. Персептрон со смещением.

Большинство алгоритмов обучения можно сформулировать более коротко, путем преобразования пороговых значений в смещения. Входной вектор (x_1, x_2,..., x_n) должен быть расширен с дополнительной единицей, и

тогда результатом будет *(n+1)*- размерный вектор (x_1, x_2,..., x_n, *1*), который будет называться расширенным входным вектором. Тогда расширенный вектор весов, относящийся к персептрону будет (w_1, w_2,..., w_n, w_{n+1}),где $w_{n+1} = -\theta$.

1.4. Задача обучения многослойного персептрона

Существует несколько формулировок задачи обучения для многослойного персептрона. Формально задачу обучения многослойного персептрона можно сформулировать как задачу приближенной интерполяции.

Формулировка I:

Дано: На вход нейронной сети подается множество обучающих пар: $\{(x^i, d^i)\}$, *i=1,...,k*, где $x^i = (x_1^i, x_2^i, \dots, x_n^k)$, $d^i = \left(d_1^i, d_2^i, \dots, d_m^i\right)$. Также нам известна погрешность $\varepsilon_1 > 0$ для обучающего множества.

Необходимо: привести весовые коэффициенты w_{ij} к такому виду, что:

$\left\|F(x^i) - d^i\right\| < \varepsilon_1 \ \forall\, i = 1..k.$

После нахождения весовых коэффициентов нейронная сеть сможет восстановить исходные данные с требуемой точностью. Тем не менее, главной для нас задачей остается хорошее восстановление данных не участвовавших в обучении и не входящих в исходное множество входных сигналов. Поэтому необходимо ввести дополнительное контрольное множество данных, чтобы выполнялось обобщающее свойства нейронной сети. Новое сформированное множество данных не будет использоваться для обучения сети, и понадобиться нам, только на этапе проверки того, как хорошо обучилась нейронная сеть, уже после того, когда мы убедимся, что мы обучили нейронную сеть должным образом. И оба множества, как контрольное, так и обучающее, должны быть достаточно разнообразны и эффективны для обучения сети и дальнейшей её проверки.

Формулировка II:

Дано: На вход сети подается два множества обучающих данных: обучающее множество $\{(x^i, d^i)\}$, $i=1,\ldots,k$, и контрольное множество $\{(x^i, d^i)\}$ $i=k+1,\ldots,k+s$, где $x^i = (x_1^i, x_2^i, \ldots, x_n^k)$, $d^i = \left(d_1^i, d_2^i, \ldots, d_m^i\right)$. Также даны погрешность $\varepsilon_1 > 0$ на обучающем множестве, погрешность $\varepsilon_2 > 0$ на контрольном множестве.

Необходимо: привести весовые коэффициенты w_{ij} к такому виду, что: $\left\|F(x^i) - d^i\right\| < \varepsilon_1, \forall i = 1, \ldots, k$ и $\left\|F(x^i) - d^i\right\| < \varepsilon_2, \forall i = k+1, \ldots, k+s$.

Для обоих формулировках $F(x)$ - функция, реализуемая нейронной сетью.

Таким образом, основной задачей, которую необходимо решить, является приведение нейронной сети к такому виду, чтобы разность ожидаемого и выходного значения нейронной сети не превосходила заданной погрешности обучения. Еще одна формулировка задачи обучения выглядит так:

Дано: множество пар обучающих данных: $\{(x^i, d^i)\}$, $i=1,\ldots,k$, также известна погрешность для обучения $\varepsilon_1 > 0$.

Найти: весовые коэффициенты w_{ij}^k такие, что:

$$E(w) = \frac{1}{2}\sum_{i=1}^{k}\left\|F\left(x^i\right) - d^i\right\|^2 = \frac{1}{2}\sum_{i=1}^{k}\left(F\left(x^i\right) - d^i\right)^2 < \frac{\varepsilon_1^2}{2}k = \varepsilon$$

Такую задачу также можно назвать задачей о минимизации целевой функции $E(w)$.

Существует большое разнообразие методов оптимизации для решения задачи обучения нейронной сети. В связи с тем, что активационные функции гладкие, мы можем использовать направленные методы оптимизации, в том числе градиентные методы.

1.5. Алгоритм обучения многослойного персептрона

Во время обучения нейронной сети, обучающие данные подаются на входной слой сети. Этот этап называется прямым ходом алгоритма обратного распространения. Во время прямого хода, каждый узел в скрытом слое получает от всех остальных узлов входного слоя значения, которые перемножаются с подходящими весовыми коэффициентами и затем суммируются. Выходные сигналы нейронов являются нелинейными преобразованиями функции активации. Аналогично, каждый выходной узел получает от всех нейронов в скрытом слое получившиеся значения, которые также перемножаются с подходящими весовыми коэффициентами и затем суммируются. Выходом каждого выходного нейрона является функция нелинейного преобразования активационной функции.

Выходные значения для последнего слоя сравниваются с идеальными выходными значениями. Идеальными выходными значениями считаются выходные обучающие данные. Ошибка между идеальным выходным значением и фактическим, полученным после прохождения сигнала через сеть рассчитывается и распространяется обратно к скрытому слою. Такое направление называется обратным ходом алгоритма обратного распространения ошибки. Ошибка используется, чтобы обновить весовые коэффициенты между узлами, например, пересчитываются веса матриц между входным-скрытым и скрытым-выходным слоями.

Алгоритм обратного распространения используется для расчета градиента целевой функции - суммы квадратов ошибок по каждой компоненте выходного сигнала. С использованием градиента целевой функции мы можем найти минимум функции ошибки в весовом пространстве с использованием градиентных методов. Найденная комбинация весов, с помощью которых минимизируется функция ошибки, считается решением задачи обучения. Так как этот способ требует

вычисления градиента функции ошибки на каждом шаге итерации, мы должны гарантировать дифференцируемость функции ошибки. Очевидно, что мы должны использовать гладкие функции активации, использование пороговых функций активации невозможно. Одна из наиболее популярных функций активации - это вещественная сигма - функция, определяемая выражением:

$$s_c(x) = \frac{1}{1 + e^{-cx}}, \tag{5}$$

Постоянная *c* может быть выбрана произвольно и ее обратная величина *1/c* называется температурным параметром в стохастических нейронных сетях. Форма сигма - функции изменяется в соответствии со значением параметра *c*, как можно увидеть на рисунке 6. График показывает форму сигма - функции при *c=1, c=2, c=3*. Наибольшее значение *c* делает сигма - функцию похожей на ступенчатую функцию, а в пределе $c \to \infty$ сигма - функция является ступенчатой (пороговой) функцией. В целях упрощения всех выражений, которые мы будем получать в дальнейшем, мы будем считать все *c = 1*. В дальнейшем мы будем называть сигма - функцию $s_1(x)$ просто $s(x)$.

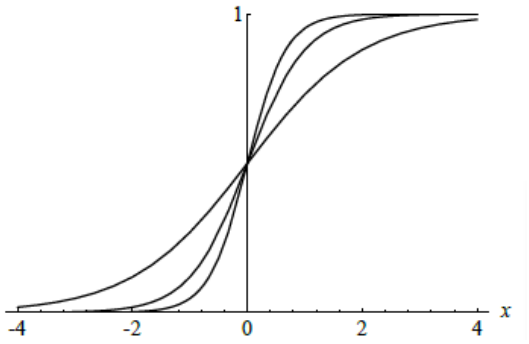

Рисунок 6. Три сигма - функции *(*$c = 1, c = 2, c = 3$*)*.

Производная от сигма - функции имеет вид:

$$\frac{d}{dx} s(x) = \frac{e^{-x}}{(1 + e^{-x})^2} = s(x)\big(1 - s(x)\big), \tag{6}$$

В случае персептрона, симметричная функция активации имеет ряд преимуществ для обучения. Альтернативой сигма - функции является биполярная (симметричная) сигма - функция, определяемая выражением:

$$S(x) = 2s(x) - 1 = \frac{1 - e^{-x}}{1 + e^{-x}}, \qquad (7)$$

Это не что иное, как гиперболический тангенс от аргумента *x/2*, форма которого показана на рисунке 7. На рисунке показаны четыре типа непрерывно - дифференцируемых функции.

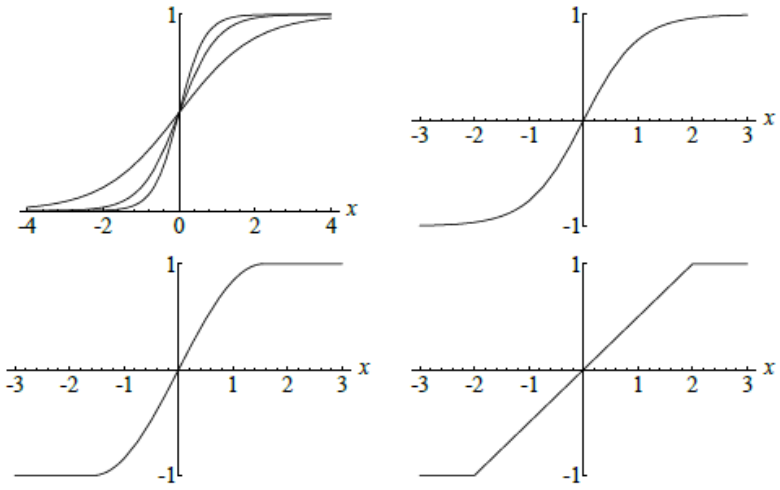

Рис. 7. Графики некоторых дифференцируемых функций активации.

Так как спуск в пространстве весов осуществляется в направлении градиента, чтобы найти минимум этой функции, важно, чтобы не существовало таких участков, на которых целевая функция полностью плоская.

Выходной диапазон сигма - функции содержит все числа строго между *0* и *1*. Оба крайних значения могут быть достигнуты только асимптотически. Используя весовые коэффициенты - w_1, w_2,..., w_n, и смещение - θ, каждый сигмоидальный нейрон по входному вектору x_1, x_2,..., x_n, вычисляет выходное значение в виде:

$$\frac{1}{1 + exp\left(\sum_{i=1}^{n} w_i x_i - \theta\right)}, \quad (8)$$

Локальные минимумы функции ошибки - это недостаток, который присущ любому направленному методу минимизации, используемому при обучении нейронных сетей. На рисунке 8 показан пример локального минимума с более высоким уровнем ошибки, чем в других местах функции. В поверхности отклика функции ошибки существует так называемая "долина", и если градиентный спуск начинается там, то алгоритм обучения остановится - происходит так называемый паралич обучения.

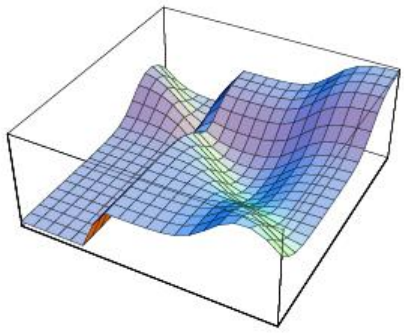

Рис. 8. Локальный минимум функции ошибки.

Локальные минимумы появляются очень часто, потому что конечные значения выходных сигналов есть значения, отличные от нуля или единицы.

Рассмотрим задачу обучения многослойного персептрона с n входными и m выходными нейронами. Также нам дан обучающий набор $\{(x_1, t_1)$, ..., $(x_p, t_p)\}$, состоящий из p упорядоченных пар m- на n- мерных векторов, которые называются входными и выходными сигналами.. Веса связей будут вещественными числами, выбранные произвольно. Когда входной сигнал x_i из обучающего множества предоставляется на вход данной сети, то сеть высчитывает другой выходной сигнал o_i, который отличается от обучающего выходного сигнала t_i. Наша цель - сделать оба сигнала t_i и o_i максимально

похожими для $i = 0,\dots, p$, используя алгоритм обучения. Точнее, мы хотим уменьшить функцию ошибки сети, определяемую как:

$$E = \frac{1}{2}\sum_{i=1}^{p} \|o_i - t_i\|^2, \tag{9}$$

до наперед заданного значения. При этом важно отметить, что задача обучения предусматривает уменьшение целевой функции, а не её минимизацию, то есть поиск локального или глобального минимума. При этом часто такую задачу также называют задачей минимизации.

Алгоритм обратного распространения используется для поиска локального минимума функции ошибок. Сеть инициализируется случайными весовыми коэффициентами. Рассчитывается градиент функции ошибки и корректируются начальные веса с учетом градиента.

Для минимизации целевой функции E, используется итерационный процесс градиентного спуска, для которого необходимо вычислить градиент .

$$\nabla E = \left(\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \dots, \frac{\partial E}{\partial w_l}\right), \tag{10}$$

Каждый вес изменяется на некоторое приращение :

$$\Delta w_i = -\gamma \frac{\partial E}{\partial w_i} \quad for\ i = 1, \dots, l, \tag{11}$$

где γ представляет собой постоянную обучения, т.е. параметр пропорциональности, который определяет длину шага на каждой итерации в отрицательном направлении градиента. Этот параметр называется коэффициентом обучения и определяет вид используемого метода спуска. Наиболее часто используется спуск с постоянным шагом либо метод наискорейшего спуска.

Алгоритм обратного распространения ошибки заключается в следующем. Мы будем рассматривать сеть с n входными нейронами, k скрытыми и m выходными. Вес между входным нейроном i и скрытым нейроном j будем обозначать ${w_{ij}}^1$. Вес между скрытым нейроном i и выходным нейроном j будем называть ${w_{ij}}^2$. Смещение - θ каждого нейрона реализовано как вес дополнительной компоненты входного сигнала. Входные веса, таким образом расширяются с помощью единичной компоненты, и то же самое делается с выходным вектором из скрытого слоя. На рисунке 9 показано, как это делается. Вес между константой *1* и скрытым нейроном j называется $w^1_{n+1,j}$, а вес между константой *1* и выходным нейроном j определяется как $w^2_{k+1,j}$.

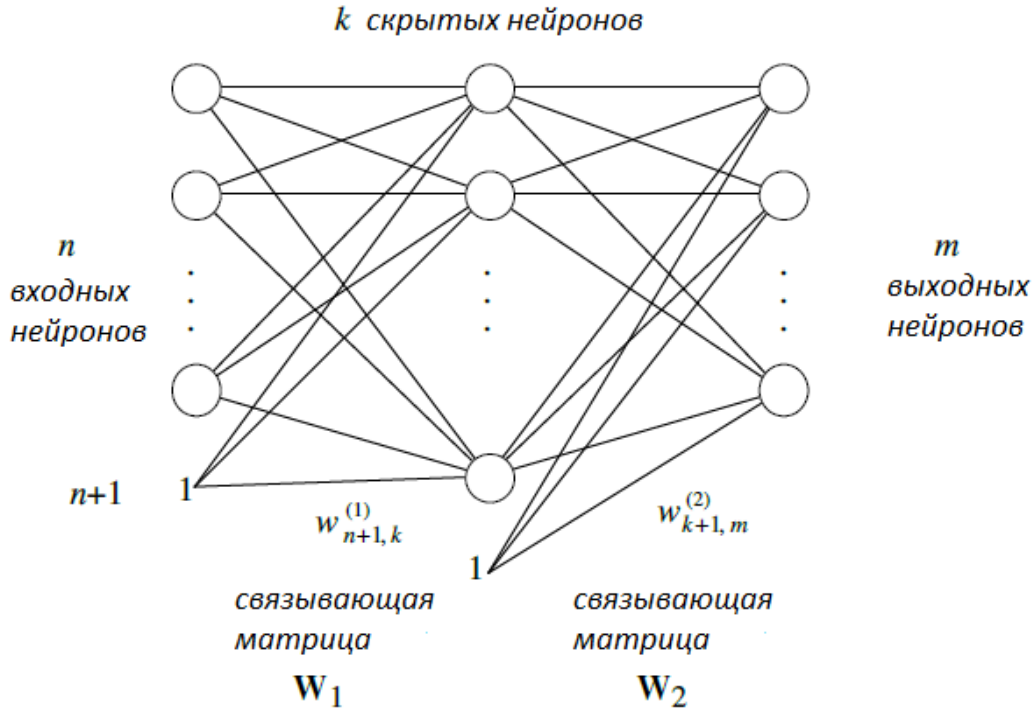

Рис. 9 Система обозначений для нейронной сети.

У нас есть $(n+1)\cdot k$ весов между входными и скрытыми нейронами и $(k+1)\cdot m$ между скрытыми и выходными нейронами. Обозначим $(n+1)\cdot k$ - мерную матрицу $\overline{W_1}$ с компонентами ${w_{ij}}^1$ для i-ой строки j-го столбца. Аналогичным образом, обозначит $(k+1)\cdot m$ - размерную матрицу $\overline{W_2}$ с компонентами ${w_{ij}}^2$. Мы используем такие обозначения, чтобы подчеркнуть,

что последняя строка обоих матриц содержит смещения вычисленных нейронов. Матрица весов без этой последней строки будет необходима на этапе обратного распространения. *N*-мерный входной вектор $\hat{o} = (o_1, o_2 \dots, o_n)$. Взвешенная сумма входных сигналов нейрона на *j*-ом скрытом слое равна:

$$net_j = \sum_{i=1}^{n+1} w_{ij}^{(1)} \widehat{o_i}, \tag{12}$$

Функция активации является сигмоидальной и выход $o_j^{(1)}$ этого нейрона выглядит следующим образом:

$$o_j^{(1)} = s\left(\sum_{i=1}^{n+1} w_{ij}^{(1)} \widehat{o_i}\right), \tag{13}$$

Значения сумматоров всех нейронов в скрытом слое может быть вычислено с помощью матричного перемножения $\hat{o}\overline{W_1}$. Вектор $o^{(1)}$, компонентами которого являются выходы скрытых нейронов, задается следующим образом:

$$\mathrm{o}^{(1)} = s(\hat{o}\overline{W_1}), \tag{14}$$

пользуясь соглашением о применении сигмоидальной функции к каждому компоненту вектора аргумента. Сумматоры нейронов в выходном слое вычисляется с использованием расширенного вектора $\widehat{o^{(1)}} = (o_1^{(1)}, \dots, o_k^{(1)}, 1)$. Выходом сети является *m* - размерный вектор $o^{(2)}$:

$$o^{(2)} = s\left(\hat{o}^{(1)}\overline{W_2}\right), \tag{15}$$

Формулы могут быть обобщены для любого числа скрытых слоев.

Для того, чтобы упростить рассуждения, мы будем говорить об одной обучающей паре *(o, t)*. Это соответствует последовательному методу обучения сети.

Алгоритм можно разложить на следующие четыре этапа:

1. Вычисления прямого хода;

2. Обратное распространения для выходного слоя;

3. Обратное распространение для скрытого слоя;

4. Обновление весов.

Алгоритм останавливается, когда значение функции ошибки становится достаточно малым.

Первый шаг: вычисление для прямого распространения.

Вектор o представлен сети. Векторы $o^{(1)}$ и $o^{(2)}$ вычислены и сохранены. Также рассчитаны производные функции активации и сохранены для каждого нейрона.

Второй шаг: обратное распространение в выходном слое.

Мы ищем первый набор частных производных $\partial E / \partial w_{ij}^{(2)}$. Путь обратного распространения от выхода сети к выходному слою j показан на рисунке 10.

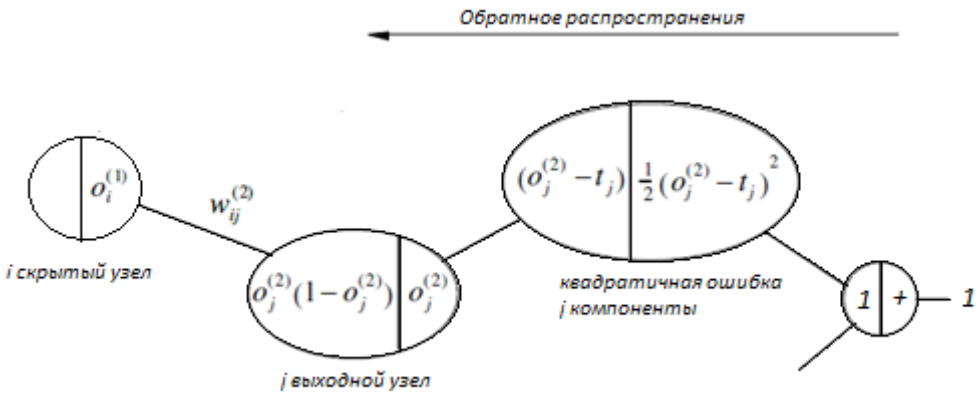

Рис. 10. Путь обратного распространения до выходного нейрона j.

Ошибка обратного распространения δ вычисляется по формуле:

$$\delta_j^{(2)} = o_j^{(2)}\left(1 - o_j^{(2)}\right)\left(o_j^{(2)} - t_j\right), \tag{16}$$

частные производные вычисляются следующим образом:

$$\frac{\partial E}{\partial w_{ij}^{(2)}} = \left[o_j^{(2)}\left(1 - o_j^{(2)}\right)\left(o_j^{(2)} - t_j\right)\right] o_i^{(1)} = \delta_j^{(2)} o_i^{(1)}, \tag{17}$$

Необходимо помнить, что на последнем шаге мы считаем веса $w_{ij}^{(2)}$ простыми переменными, а их выходы $o_i^{(1)}$ постоянными.

На входном конце связи с весом w_{ij} мы имеем $o_i^{(1)}$ и на выходной конце ошибку обратного распространения $\delta_j^{(2)}$.

Третий шаг: обратное распространение в скрытом слое.

Теперь мы считаем частные производные $\partial E \Big/ \partial w_{ij}^{(1)}$. Каждый нейрон j в скрытом слое подсоединен к нейрону q выходного слоя с весом $w_{jq}^{(2)}$ для *q=1, .., m*. Ошибка обратного распространения до нейрона j в скрытом слое должна быть рассчитана с учетом всех возможных обратных связей, как показано на рисунке 11. Ошибка обратного распространения тогда имеет вид:

$$\delta_j^{(1)} = o_j^{(1)}\left(1 - o_j^{(1)}\right)\sum_{q=1}^{m} w_{jq}^{(2)} \delta_q^{(2)}, \tag{18}$$

Таким образом частную производную мы находим по формуле:

$$\frac{\partial E}{\partial w_{ij}^{(1)}} = \delta_j^{(1)} o_i, \tag{19}$$

Ошибка обратного распространения рассчитывается таким же образом для всех скрытых слоев, и выражение для частной производной *E* сохраняет такую же аналитическую форму.

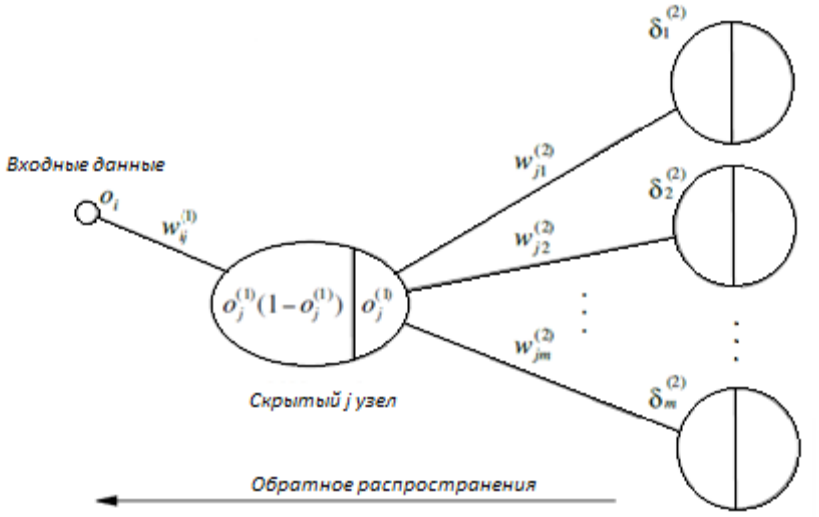

Рис.11. Все пути до входной i-ой стороны.

Четвертый шаг: изменение весов.

После вычисления всех частных производных сети веса изменяются в направлении антиградиента. Коэффициент обучения γ определяет длину шага поправки. Поправки весов определяются по формуле:

$$\Delta w_{ij}^{(2)} = -\gamma o_i^{(1)} \delta_j^{(2)}, \text{для } i = 1, \dots, k+1; j = 1, \dots, m, \tag{20}$$

или

$$\Delta w_{ij}^{(1)} = -\gamma o_i \delta_j^{(1)}, \text{для } i = 1, \dots, n+1; j = 1, \dots, k, \tag{21}$$

где $o_{n+1} = o_{k+1}^{(1)} = 1$. Очень важно добавить поправки весов только после того, как ошибки обратного распространению будут вычислены для всех нейронов в сети.

1.6. Проблемы обучения сети

Локальный минимум.

На практике существуют некоторые особенности на поверхности отклика функции суммы квадратов ошибок, такие как "овраги" и "плато", что может представлять собой трудности для минимизации целевой функции. Например, две функции обработки ошибок, показанные на рисунке 12, не имеют локальных минимумов . Тем не менее, функция с левой стороны, как ожидается, будет сложнее для оптимизации с методом градиентного спуска.

По оси абсцисс отложено значение одного параметра, а по оси ординат соответствует функция ошибки. Не смотря на то, что ни одна из этих функций не содержит локальные минимумы, функция слева будет менее пригодна для градиентной оптимизации спуска из-за плоских участков.

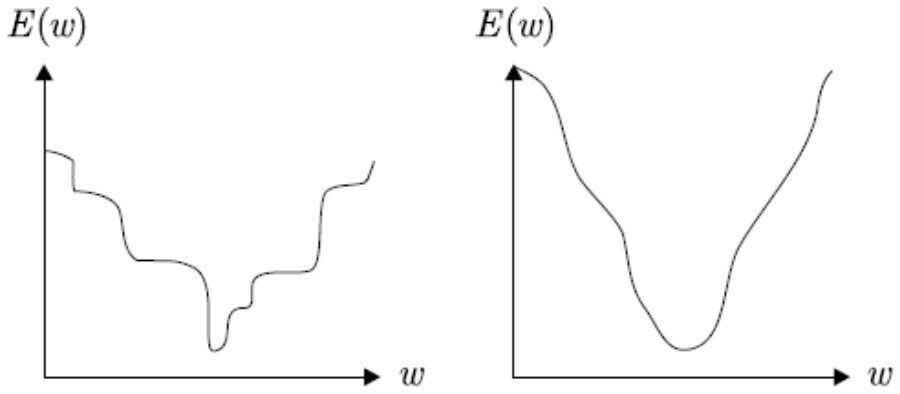

Рис. 12. Две функции ошибки одной размерности.

Обобщение и переобучение.

Свойство обобщения показывает то, насколько хорошо сеть выполняет свои функции на новых данных, которые не участвовали в процессе обучения. Модель сети обучается на заранее известных обучающих данных и обобщение соответствует ожидаемой эффективности модели по новым данным для проверки.

Математически, цель обучения для многослойного персептрона может быть сформулирована, как минимизация функции "стоимости":

$$E_{true} = \int_{x,d} e(f(x.w), d)p(x, d)dxdd, \tag{22}$$

где e - локальная функция стоимости, f - функция реализуемая с помощью многослойного персептрона, x - входящий сигнал сети, d - это ожидаемый, w - соответствующие веса в сети, p - вероятность распределения. Целью обучения является оптимизация параметров w, что сводит функцию E_{true} к минимуму:

$$\widehat{w} = argmin_w \int_{x,d} e(f(x.w), d)p(x, d)dxdd, \tag{23}$$

E_{true} - ошибка обобщения, т.е. ожидаемая производительность многослойного персептрона на новых заранее неизвестных примерах входных данных, выбранных произвольно из $p(x, d)$. На практике $p(x, d)$ неизвестная величина. Вместо этого, нам даны обучающие пары данных, и функция, которую необходимо минимизировать. Она называется эмпирической функцией или функцией ошибки обучения:

$$E = \sum_{p=1}^{N_p} e(x_p, d_p), \tag{24}$$

Очень важный вопрос состоит в том, чтобы понять, как хорошо модель сети натренирована для минимизации функции E обобщенно (т.е. насколько мала функция E_{true}). Это связано с тем, что малая величина функции E (производительность на обучающей выборке) не означает малость и функции E_{true}(ожидаемая производительность на новых примерах).

Многослойный персептрон реализует функцию отображения вектора входных данных в желаемый выходной вектор значений. Это отображение, как правило "сглаживается" (в смысле, определенном природой функции активации, топологией сети, и алгоритмом обучения) и допускает интерполяцию между точками обучения. Рассмотрим простой случай входного сигнала с единичной размерностью, как показано на рисунке 13. Тренировочные примеры, отмеченные крестом, содержат шум. Истинным для функции отображения, может быть то, что показано в середине графика. Однако, без какой-либо контролирующей схемы, многослойный персептрон может серьезно недообучиться (левый график на рисунке 13) или переобучиться (правый график на рисунке 13). Заметим, что средняя ошибка на тренировочных данных является самой высокой для недообучения персептрона на рисунке 13, и самой низкой для переобученного персептрона. Для случая переобучения, ошибка на обучающих данных может быть очень низкой, в то время как ошибка на тестируемых образцах может быть очень высокой (рассматривая тестируемые данные между обучающими точками на графике с переобучением), т.е. для данной многослойной сети, по мере дальнейшего обучения между "правильными" точками, производительность обобщения может уменьшиться. В случае недообучения, оценщик многослойной сети дает оценки, которые имеют большое смещение, но низкую дисперсию (оценщик называется смещенным, если, в среднем, оценочная стоимость отличается от ожидаемого значения). В случае переобучения, смещение оценщика низкое, но дисперсия высокая. Таким образом, существует оптимальный вариант между двумя крайностями.

Степень, с которой сеть переобучается, возможно связана с количеством тренировочных пар и количеством параметров в модели сети. В общем случае, с фиксированным количеством тренировочных пар, переобучение может возникнуть, когда нейронная сеть имеет слишком много параметров (слишком много степеней свободы). Рисунок 14 иллюстрирует идею с использованием полиноминальной аппроксимации.

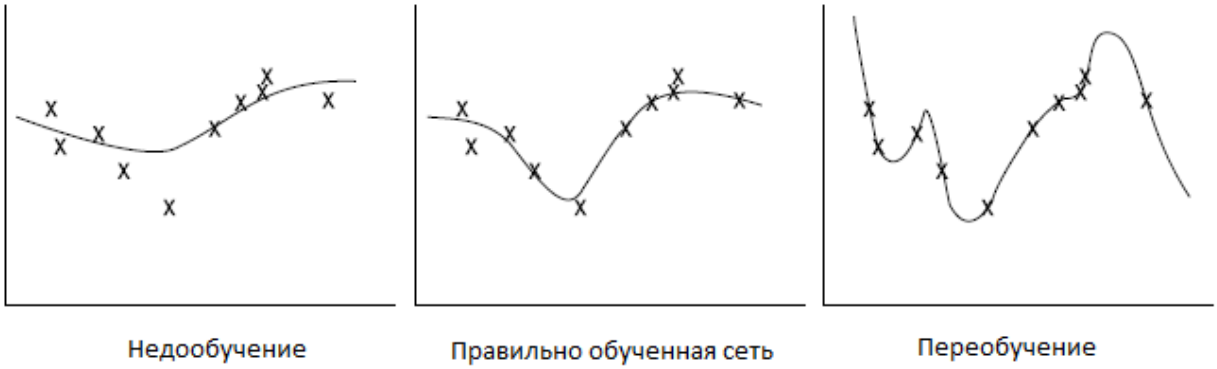

Рисунок 13. Недообучение, правильное обучение и переобучение.

Обучающий набор данных содержит 21 точку относительно уравнения $y = \sin\left(\frac{x}{3}\right) + \upsilon$, где υ - это равномерно-распределенная случайная величина между -0.25 и 0.25. Уравнение было оценено в точках 0, 1, 2, ..., 20. Этот набор данных был затем использован, чтобы соответствовать полиномиальным моделям с порядком от 2 до 20. Для порядка 2, приближение не очень хорошее, как показано на рисунке 14. А для порядка 10 приближение достаточно хорошее. Тем не менее, поскольку порядок (и количество параметров) возрастает, значительность переобучения очевидна.

В случае порядка 20, аппроксимирующая функция подходит для тренировочных данных очень хорошо, однако интерполяция между обучающими точками очень плохая.

На рисунке 15 показаны результаты использования многослойного персептрона для того, чтобы аппроксимировать те же обучающие данные. Что касается полиномиального случая , самая маленькая сеть с одним скрытым слоем (4 веса, в том числе вес смещения) не аппроксимирует данные хорошо. С помощью двух скрытых слоев (7 весов) приближение становится достаточно хорошим. Однако, в отличии от сети с 10 скрытыми слоями (31 вес) и 50 скрытыми слоями (151 вес) обучение также может привести к достаточно хорошему приближению.

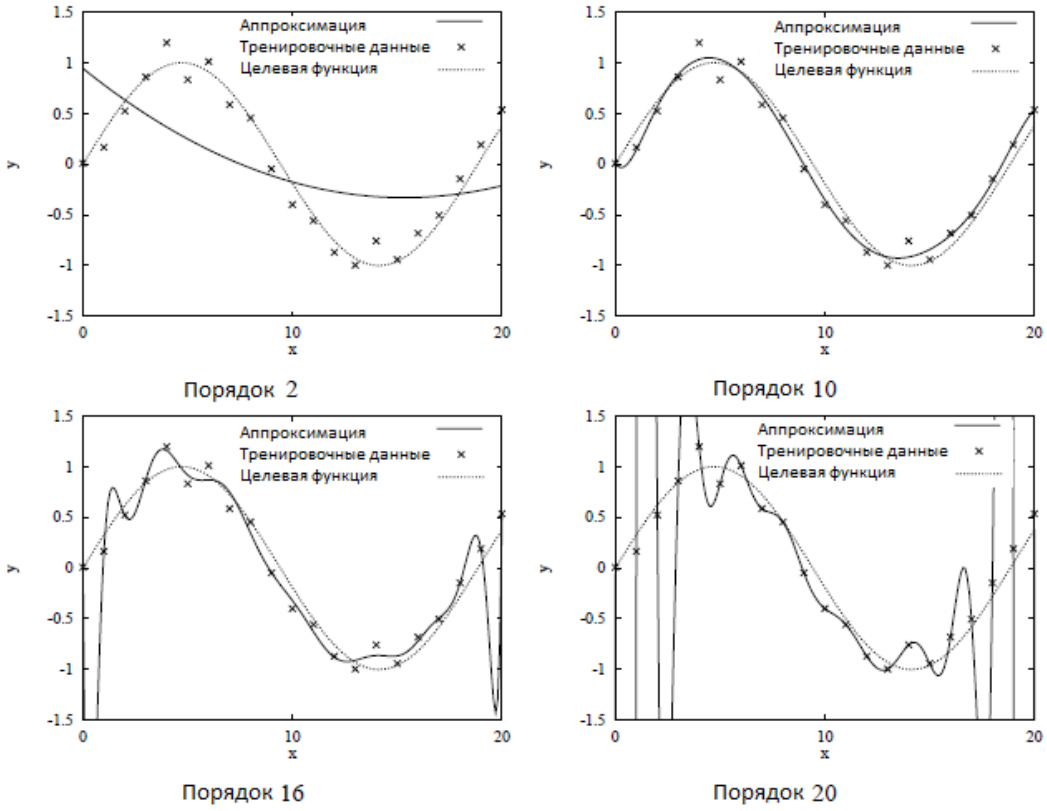

Рисунок 14. Полиномиальная аппроксимация на примере функции $y = \sin\left(\frac{x}{3}\right) + \upsilon$, где порядок модели увеличивается от 2 до 20.

Следовательно, для конкретного (очень простого) примера, сеть многослойного персептрона, обученная с помощью обратного распространения не приводит к переобучению в значительной степени, даже если количество параметров превышает в семь раз количество точек данных. Это, конечно, верно, что переобучение может быть серьезной проблемой для многослойного персептрона. Тем не менее, этот пример подчеркивает возможность того, что многослойный персептрон, обученный обратным распространением ошибки может быть смешен в сторону более плавной аппроксимации.

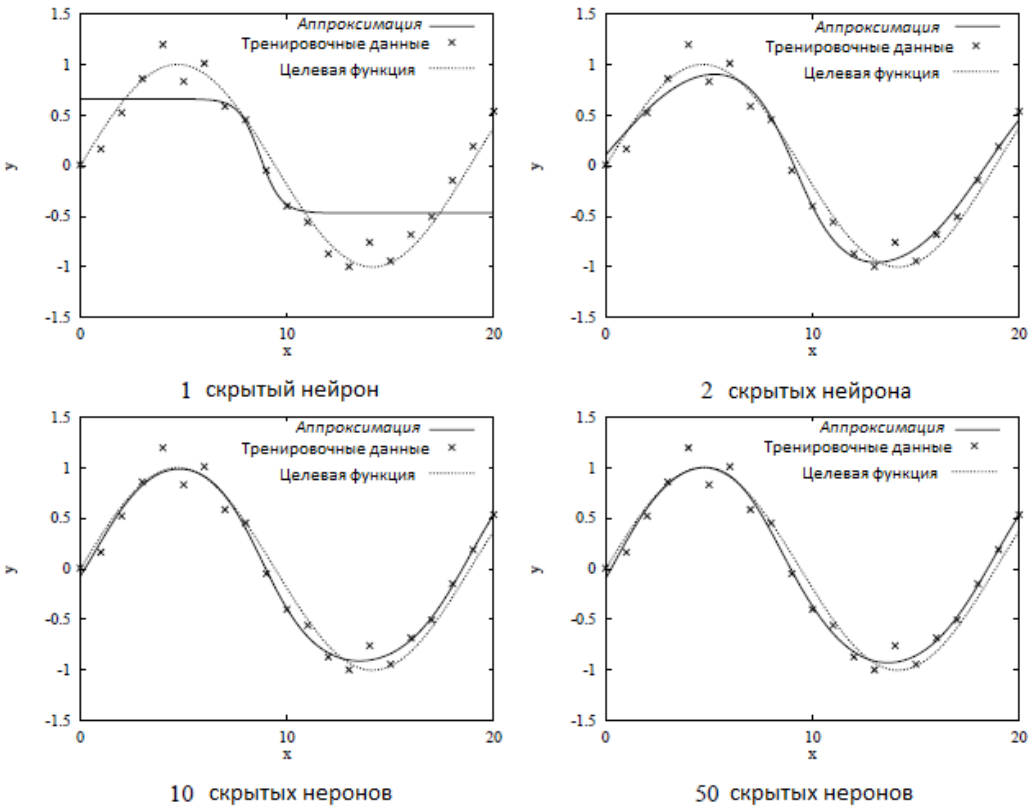

Рисунок 15. Интерполяция многослойного персептрона на примере функции $y = \sin\left(\frac{x}{3}\right) + v$.

1.7. Проблема выбора количества скрытых слоев и количества нейронов в скрытом слое

Создание архитектуры нейронной сети означает указание количества слоев каждого типа и количества нейронов в каждом из этих слоев.

Входной слой - один из самых простых слоев, который имеет абсолютно любая нейронная сеть без исключений. Количество нейронов, содержащихся в этом слое полностью и однозначно определяется размерностью входного сигнала. Важно отметить, что входной слой фактически не выполняет никаких вычислительных функций и может считаться слоем разделителей компонент входного сигнала. Некоторые

конфигурации нейронных сетей добавляют один дополнительный узел для смещения.

Выбор числа нейронов для выходного слоя очень схож подходом для входного слоя. Определение количества нейронов очень просто. Оно полностью определяется размерностью выходного сигнала.

Существует несколько правил, которые позволяют установить количество слоев и размер нейронного слоя для обоих входных и выходных слоев. Основной вопрос архитектуры многослойного персептрона касается скрытых слоев. Сколько скрытых слоев должно быть в нейронной сети? Хорошо, если задача линейно разделима, тогда не нужно каких-либо скрытых слоев вообще. Одна из проблем в этой теме, по которой существует консенсус, т.е. разница производительности от добавления дополнительных скрытых слоев - ситуация, в которой повышается производительность с добавлением второго (третьего, четвертого, и т.д.) скрытого слоя очень маловероятна. Один скрытый слой достаточен для подавляющего большинства задач.

А вот для определения числа нейронов в скрытом или скрытых слоях существуют эмпирически-обоснованные правила. В общем и целом большинство полагаются на этот "оптимальный размер скрытого слоя, который основывается на количестве нейронов во входном и выходном слое". Также считается, что количество нейронов в скрытом слое должно быть в несколько раз меньше количества обучающих примеров, при условии наличия избыточности обучающих данных.

Суммируя все выше сказанное, для большинства задач можно было бы, вероятно, получить достойную производительность установив конфигурацию скрытого слоя , используя только два правила :

1) количество скрытых слое равно единице;

2) количество нейронов в этом слое является средним между количеством нейронов во входном и выходном слоях и быть в несколько раз меньше

количества обучающих примеров, при условии наличия избыточности обучающих данных.

Для того, чтобы нейронная сеть обучилась правильно на заданной выборке данных с наибольшей точностью, необходимо подобрать это число нейронов наиболее оптимально. Их должно быть не много, но и достаточно для хорошей аппроксимации функции в пространстве синоптических весов. В случае, если количество нейронов в скрытом слое недостаточно, то задача обучения на обучающем множестве не может быть решена. В случае, если нейронов слишком много, сеть может обучиться избыточно. При этом фактически происходит запоминание различных шумов и погрешностей обучающих данных. При этом процесс обучения существенно замедляется из-за большего количества весовых коэффициентов сети. Поэтому для настройки оптимальной архитектуры сети применяются различные методы. Такие способы делятся на две группы: алгоритмы сокращения и конструктивные алгоритмы.

Глава 2. Нейронная сеть линейной аппроксимации

2.1. Сеть линейной аппроксимации

Сеть линейной аппроксимации состоит из двух слоев нейронов обеспечивающих разбиение входных обучающих данных n-мерными симплексами и их аппроксимацию гиперплоскостями на каждом симплексе (рис. 16). Сеть предназначена для решения задач прогнозирования, распознавания и классификации образов. Данная сеть состоит из входного слоя разделителей сигнала, двух скрытых слоев и выходного слоя, а также модулятора аппроксимации. В процессе обучения с учителем нейроны первого слоя разделяют множество обучающих входных данных на n-мерные симплексы. Нейроны первого слоя, реализующие грани одного симплекса

соединены связями с ненулевыми весовыми коэффициентами с одним нейроном второго слоя. Для каждого симплекса соответствующий нейрон второго слоя реализует гиперплоскость, аппроксимирующую выходные обучающие сигналы. Архитектура сети линейной аппроксимации изображена на рисунке.

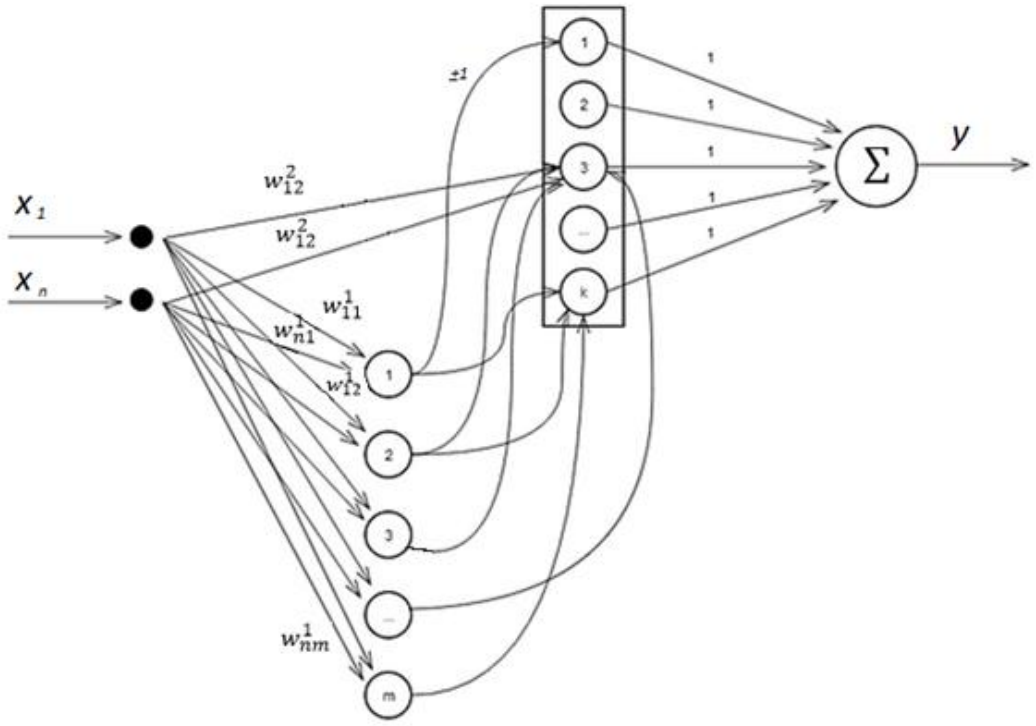

Рис.16. Сеть линейной аппроксимации.

Количество разделителей сигналов равно размерности входного сигнала. Разделители сигнала обеспечивают передачу сигнала на первый слой и препятствуют прохождению входного сигнала на второй слой до получения разрешающего сигнала модулятора аппроксимации. Все разделители сигнала связаны с модулятором аппроксимации. В качестве разделителей сигнала могут использоваться нейроны с линейной функцией активации. Выходной слой содержит нейроны с линейной функцией активации $f(s) = s$, количество которых равно размерности выходного вектора. В данной работе рассматривается вариант сети с один выходным нейроном. Для задачи с размерностью выходного сигнала больше единицы сеть линейной аппроксимации должна содержать несколько вторых

аппроксимационных слоев. Их количество должно быть равно размерности выходного сигнала.

Первый слой отвечает за разделение входных данных в n-мерном пространстве входных сигналов $n-1$-мерными гиперплоскостями на n - мерные симплексы. Первый слой состоит из нейронов с пороговой функцией активации. Веса связей нейронов первого слоя со слоем разделителей сигнала, а также значения порогов определяются в процессе обучения. Группа нейронов первого слоя, реализующих грани одного n - мерного симплекса называется симплексной группой нейронов. Второй слой состоит из пар связанных нейронов. Первый нейрон из пары нейронов второго слоя имеет полулинейную функцию активации и связан с нейронами первого слоя. Эти нейроны обеспечивают определение симплекса внутри которого находится вектор входного сигнала. Второй нейрон каждой нейронной пары второго слоя имеет линейную функцию активации и связан с входным слоем разделителей сигнала. Он обеспечивает линейную аппроксимацию обучающих пар (x^i, d^i) в $(n+1)$-мерном пространстве n - мерными гиперплоскостями. Каждая пара нейронов второго слоя соответствует одной симплексной группе нейронов первого слоя. Связи вторых нейронов нейронных пар второго слоя с элементами входного слоя разделителей компонент входного сигнала имею временную задержку, позволяющую задержать передачу сигнала со входного слоя на второй слой до окончания процесса конкуренции. Этой временной задержкой управляет модулятор аппроксимации, который передает управляющий сигнал, разрешающий прохождение входного сигнала на второй слой только после окончание конкуренции. Вектор весовых коэффициентов связей каждого второго нейрона нейронных пар второго слоя соответствует одной аппроксимирующей гиперплоскости и определяется при обучении сети. Такая гиперплоскость аппроксимирует входные сигналы, находящиеся в n - мерном симплексе, соответствующем нейронной паре второго слоя.

Кроме того, каждый первый нейрон нейронной пары второго слоя связан обратными связями со всеми остальными первыми нейронами нейронных пар второго слоя. Веса обратных связей определяются следующим образом (аналогично сети Хемминга, см. например [2]):

$$v_{ij} = \begin{cases} 1, & i = j \\ -\frac{1}{p-1} + a_j, & i \neq j \end{cases}, \qquad (27)$$

Здесь p – количество нейронных пар второго слоя, $a_j > 0$ – небольшое произвольное число. После прохождения сигнала с первого слоя начинается итерационный процесс, в результате которого обнуляются выходные значения всех первых нейронов нейронных пар второго слоя, кроме одного нейрона-победителя. Этим нейроном является нейрон с наибольшим значением взвешенной суммы компонент выходного сигнала первого слоя. Выходные значения остальных первых нейронов второго слоя обнуляются до подачи на вход сети нового сигнала. По окончании итерационного конкурентного процесса на вход вторых нейронов нейронных пар второго слоя передается сигнал с входного слоя. Процесс передачи сигнала управляется модулятора аппроксимации.

Третий слой состоит из одного линейного нейрона, обеспечивающего передачу одного ненулевого сигнала от нейрона второго слоя на выход сети. Он соединен прямыми связями со всеми вторыми нейронами нейронных пар второго слоя. Эти связи имеют единичные весовые коэффициенты.

При программной реализации сети линейной аппроксимации вместо пар нейронов во втором слое могут использоваться нейроны с полулинейной функцией активации без обратных связей. Порог полулинейной функции активации принимается нулевым, а режим функционирования этих нейронов определяется сигналом модулятора аппроксимации. При этом процесс конкуренции может моделироваться алгоритмом сортировки который обеспечивает определение нейрона с максимальным выходным значением.

2.2. Обучение сети линейной аппроксимации

Задача обучения сети линейной аппроксимации ставится аналогично задаче обучения многослойного персептрона.

Дано: 2 набора обучающих пар: $\{(x^i,\ d^i)\}$, $i=1,\ldots,k$, $\{(x^i,\ d^i)\}$, $i=k+1,\ \ldots,\ k+s$, где вектор

$$x^i = (x_1^i, x_2^i, \ldots,\ x_n^k)\ ,$$
$$d^i = \left(d_1^i, d_2^i, \ldots,\ d_n^k\right).$$

Погрешность ε_1 *на обучающем множестве и* погрешность ε_2 на контрольном множестве.

Необходимо: подобрать значение весовых коэффициентов w_{ij} таким образом, чтобы:

$$\left\|F(x^i) - d^i\right\| < \varepsilon_1 \text{ и } \left\|F(x^i) - d^i\right\| < \varepsilon_2.$$

Для обучения сети линейной аппроксимации предлагается следующий пакетный метод обучения сети линейной аппроксимации. Этот алгоритм является конструктивным, то есть одновременно с определением весовых коэффициентов нейронных связей в сеть добавляются новые нейроны. Таким образом количество нейронов в первом и втором слоях настраивается автоматически в процессе ее обучения. Первоначально весовые коэффициенты всех нейронов обнулены, за исключением весовых коэффициентов обратных связей нейронов второго слоя и единичных весовых коэффициентов выходного нейрона. В процессе обучения эти веса получают ненулевые значения по алгоритму, изложенному ниже. Этот процесс мы будем называть добавлением нового нейрона.

Алгоритм обучения сети линейной аппроксимации следующий:

1 этап: Предварительное обучение.

Выбираются $n+1$ векторов x^i, не принадлежащих одной гиперплоскости. Необходимо, чтобы при предварительном обучении обучения все входные вектора обучающего множества находились внутри n - мерного симплекса,

вершинами которого являются указанные $n+1$ векторов. В случае отсутствия таких обучающих данных, они вводятся искусственно. При большой размерности входного сигнала и сложностях в выборе указанных векторов допускается использование для предварительного обучения n - мерного прямоугольного параллелепипеда. Вершинами этого параллелепипеда являются $n+2$ вектора, которые определяются из максимумов модулей по каждой компоненте всех векторов обучающего множества. Затем определяются веса связей $n+1$ нейронов первого слоя с элементами входного слоя разделителей компонент входного сигнала. Для *i-го* нейрона первого слоя вектор весовых коэффициентов его связей с элементами входного слоя разделителей $w_i^1 = (w_{1i}^1, \dots, w_{ni}^1)$ определяются из системы линейных алгебраических уравнений (грань симплекса строится по n точкам - векторам x^i)[11,15]:

$$\begin{cases} w_{1i}^1 x_1^1 + w_{2i}^1 x_2^1 + w_{3i}^1 x_3^1 + \cdots + w_{ni}^1 x_n^1 = 1 \\ w_{1i}^1 x_1^2 + w_{2i}^1 x_2^2 + w_{3i}^1 x_3^2 + \cdots + w_{ni}^1 x_n^2 = 1 \\ \dots \\ w_{1i}^1 x_1^n + w_{2i}^1 x_2^n + w_{3i}^1 x_3^n + \cdots + w_{ni}^1 x_n^n = 1 \end{cases}, \tag{28}$$

Здесь вектор весов $w_i^1 = (w_{1i}^1, \dots, w_{ni}^1)$ *i-го* нейрона является вектором коэффициентов гиперплоскости

$$w_{1i}^1 x_1 + w_{2i}^1 x_2 + \cdots + w_{ni}^1 x_n - 1 = 0, \tag{29}$$

являющейся гранью n - мерного симплекса, внутри которого находятся все входные вектора обучающего множества. Порогом каждого нейрона первого слоя является -1. Для выбора n векторов необходимо из набора n+1 векторов каждый раз удалять один вектор. Удаляемый вектор x^j используется для определения весов связей 1-го нейрона второго слоя (первый нейрон нейронной пары) со всеми нейронами первого слоя по правилу

$$\begin{cases} 1, & \text{если } w_{1i}^1 x_1^j + \cdots + w_{ni}^1 x_n^j - 1 > 0 \\ & \\ -1, & \text{если } w_{1i}^1 x_1^j + \cdots + w_{ni}^1 x_n^j - 1 < 0 \end{cases}, \tag{30}$$

$$w_{i1} = \begin{cases} 1, & \text{если } w_{1i}^1 x_1^j + \cdots + w_{ni}^1 x_n^j - 1 > 0 \\ & \\ -1, & \text{если } w_{1i}^1 x_1^j + \cdots + w_{ni}^1 x_n^j - 1 < 0 \end{cases}, \tag{31}$$

После этого добавляется (инициализируется) один нейрон (второй нейрон нейронной пары) второго слоя, соответствующий симплексной группе. Весовые коэффициенты связей этого нейрона с элементами слоя разделителей компонент входного сигнала определяются из системы линейных алгебраических уравнений (гиперплоскость строится по n точкам - векторам (x^i, d^i)):

$$\begin{cases} v_1 x_1^1 + \cdots + v_n x_n^1 + v_{n+1}\, d^1 = 1 \\ v_1 x_1^2 + \cdots + v_n x_n^2 + v_{n+1}\, d^2 = 1 \\ \cdots \\ v_1 x_1^n + \cdots + v_n x_n^n + v_{n+1}\, d^n = 1 \end{cases}, \tag{32}$$

$$w_i^2 = -\frac{v_i}{v_{n+1}},\ \ i = 1..n, \tag{33}$$

Здесь вектор весов $w^2 = (w_1^2, \ldots, w_n^2)$ первого нейрона второго слоя является вектором коэффициентов решающей гиперплоскости

$$w_i^2 x_1 + \cdots + w_n^2 x_n + d = z, \tag{34}$$

Порогом каждого нейрона первого слоя является $d = \frac{1}{v_{n+1}}$.

2 этап: Основное обучение в пакетном режиме.

Основной этап обучения в пакетном режиме представляет собой итерационный процесс на каждом шаге которого выполняются следующие действия.

1) На вход сети подаются все входные обучающие сигналы, один за другим. Для каждого обучающего сигнала вычисляется модуль ошибки. Затем определяется обучающий пример с максимальным значением модуля ошибки. Этот пример x^j далее используется для обучения сети.

2) На вход сети подается обучающий входной вектор x^j, определенный на шаге 1 и определяем в каком n-мерном симплексе он находится, исходя из максимального выходного значения нейронов второго скрытого слоя. Вычисляется выходное значение сети и сравнивается с эталонным значением d^j. В случае, если выходное значение удовлетворяет условию $\left\|F(x^j) - dj\right\| < \varepsilon 1$, весовые коэффициенты не изменяются и обучение завершается.

3) В случае, если условие на шаге 2 не выполнено, добавляем еще n нейронов в первый слой и один нейрон во второй слой. Таким образом, n-мерный симплекс, в который попал вектор x^j разделяется этим вектором на $n+1$ новый симплекс. От прежнего симплекса каждый новый симплекс получит одну грань. Соответственно все нейроны первого скрытого слоя с инициализированными весами сохраняют свои весовые коэффициенты, соответствующие связям с входным слоем разделителей входного сигнала. Далее рассчитываем весовые коэффициенты у добавленных нейронов по правилам, изложенным при описании предварительного обучения.

3. Вычислительный эксперимент

3.1. Задача аппроксимации функции двух переменных

Для проведения вычислительного эксперимента двухслойный персептрон и сеть линейной аппроксимации были реализованы в виде компьютерной программы в среде Matlab.

Для проверки работоспособности нейронных сетей и проведения вычислительного эксперимента рассмотрим задачу аппроксимации для функции двух переменных:

$$f(x_1, x_2) = \sin(x_1) + x_2, \tag{35}$$

в квадрате [0,10]x[0,10]. Данных для обучения мы выбрали представительным обучающем множеством, а остальные примеры использовали для тестирования работы сети. Погрешность ошибки обучения для обучающего множества была выбрана равной ε_1= 0,001, а для тестового $\varepsilon_2 = 0{,}01$.

Результат аппроксимации нейронной сетью функции изображен на рисунке.

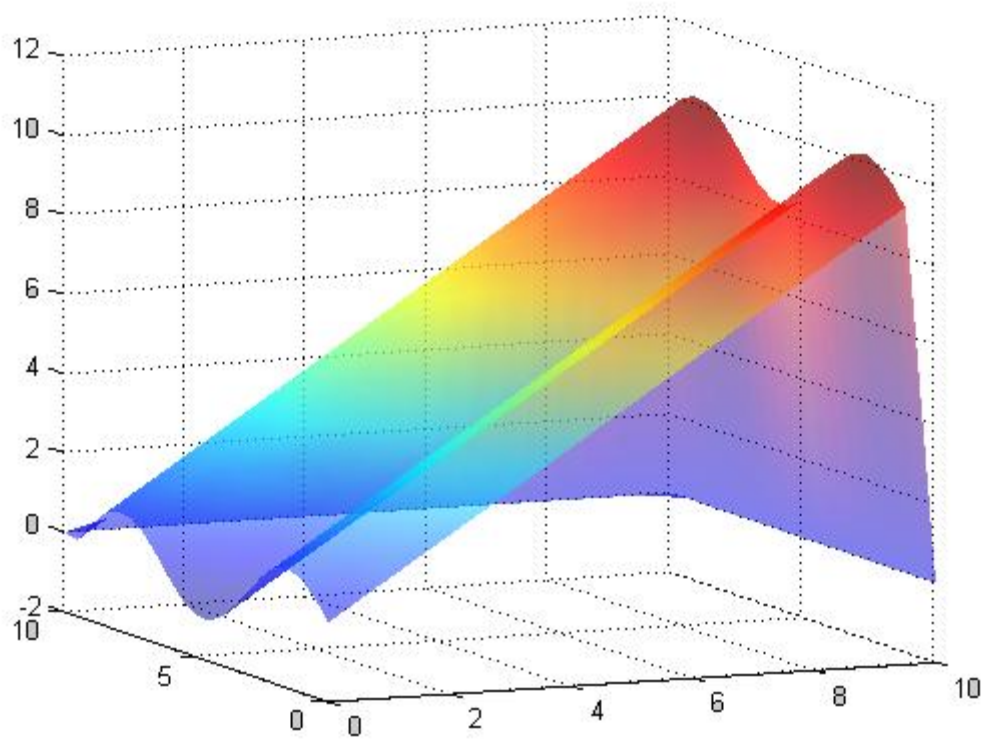

Рис.17. Результат работы персептрона для задачи аппроксимации функции двух переменных.

Сеть обучилась за 34 секунды в течении 200 итераций. В качестве множества обучающих входных сигналов использовалась равномерная сетка

на квадрате [0,10]x[0,10] с шагом $h = 0.3$. Количество обучающих примеров было равно 35. В качестве контрольного множества входных данных использовалась та же сетка, сдвинутая вдоль осей Ox и Oy на $p = 0.1$. Точность, с которой была обучена нейронная сеть для решения поставленной задачи была определен равной $e = 0.0001$.

На следующем графике можно проследить как изменялась квадратичная ошибка обучения с течением времени:

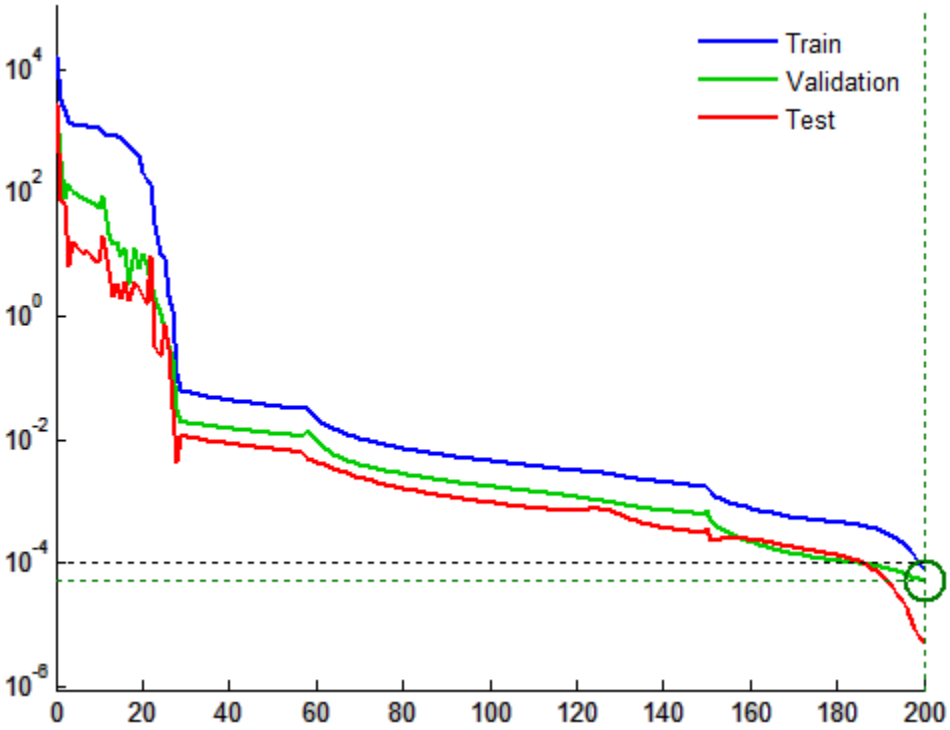

Рис. 18. Квадратичная ошибка обучения для задачи аппроксимации функции двух переменных.

На графике ошибки обучения видно. что при проведении обучения возникали многочисленные локальные минимумы целевой функции с которым алгоритм обучения успешно справился, что предотвратило паралич обучения. Проведено обучение обеих нейронных сетей. Обучение завершилось успешно со следующими параметрами из таблицы 1.

	Многослойный персептрон	Сеть линейной аппроксимации
Количество нейронов в скрытом слое	7	72
Время обучения сети	34	41

Таблица.1. Результаты вычислительного эксперимента.

3.2. Задача определения сортов вин

Второй эксперимент, который мы выбрали для проверки работоспособности нейронных сетей - это задача распознавание сортов вин. Данные, которые мы использовали в эксперименте представляют собой результаты химического анализа вина, выращенного в одном регионе Италии, и имеющие разное происхождение. Анализ определил 13 компонент, составляющие каждое из трех видов вина. Известно, что первоначально набор данных имел около 30 компонент, но мы решили сократить их до 13. Эти 13 компонент вина следующие:

1) Алкоголь
2) Яблочная кислота
3) Ясень
4) Щелочность золы
5) Магний
6) Всего фенолы
7) Флавоноиды
8) Нонфлавоноиды, фенолы
9) Проанфоцианинс
10) Интенсивность цвета
11) Оттенок
12) OD280/OD315 разбавленное вино
13) Пролина

Матрица данных для обучения нейронной сети выглядит таким образом:

1,14.23,1.71,2.43,15.6,127,2.8,3.06,.28,2.29,5.64,1.04,3.92,1065
1,13.16,2.36,2.67,18.6,101,2.8,3.24,.3,2.81,5.68,1.03,3.17,1185
1,13.24,2.59,2.87,21,118,2.8,2.69,.39,1.82,4.32,1.04,2.93,735

1,14.39,1.87,2.45,14.6,96,2.5,2.52,.3,1.98,5.25,1.02,3.58,1290
1,14.83,1.64,2.17,14,97,2.8,2.98,.29,1.98,5.2,1.08,2.85,1045
1,14.1,2.16,2.3,18,105,2.95,3.32,.22,2.38,5.75,1.25,3.17,1510
1,13.75,1.73,2.41,16,89,2.6,2.76,.29,1.81,5.6,1.15,2.9,1320
1,14.38,1.87,2.38,12,102,3.3,3.64,.29,2.96,7.5,1.2,3,1547
1,14.3,1.92,2.72,20,120,2.8,3.14,.33,1.97,6.2,1.07,2.65,1280
1,14.19,1.59,2.48,16.5,108,3.3,3.93,.32,1.86,8.7,1.23,2.82,1680
1,14.06,1.63,2.28,16,126,3,3.17,.24,2.1,5.65,1.09,3.71,780
1,13.71,1.86,2.36,16.6,101,2.61,2.88,.27,1.69,3.8,1.11,4,1035
1,13.5,1.81,2.61,20,96,2.53,2.61,.28,1.66,3.52,1.12,3.82,845
1,13.39,1.77,2.62,16.1,93,2.85,2.94,.34,1.45,4.8,.92,3.22,1195
1,13.87,1.9,2.8,19.4,107,2.95,2.97,.37,1.76,4.5,1.25,3.4,915
1,13.73,1.5,2.7,22.5,101,3,3.25,.29,2.38,5.7,1.19,2.71,1285
1,13.68,1.83,2.36,17.2,104,2.42,2.69,.42,1.97,3.84,1.23,2.87,990
1,13.51,1.8,2.65,19,110,2.35,2.53,.29,1.54,4.2,1.1,2.87,1095
1,13.28,1.64,2.84,15.5,110,2.6,2.68,.34,1.36,4.6,1.09,2.78,880
1,13.07,1.5,2.1,15.5,98,2.4,2.64,.28,1.37,3.7,1.18,2.69,1020
1,13.56,1.71,2.31,16.2,117,3.15,3.29,.34,2.34,6.13,.95,3.38,795
1,13.88,1.89,2.59,15,101,3.25,3.56,.17,1.7,5.43,.88,3.56,1095
1,13.05,1.77,2.1,17,107,3,3,.28,2.03,5.04,.88,3.35,885
1,14.38,3.59,2.28,16,102,3.25,3.17,.27,2.19,4.9,1.04,3.44,1065
1,14.1,2.02,2.4,18.8,103,2.75,2.92,.32,2.38,6.2,1.07,2.75,1060
1,13.05,1.73,2.04,12.4,92,2.72,3.27,.17,2.91,7.2,1.12,2.91,1150
1,13.82,1.75,2.42,14,111,3.88,3.74,.32,1.87,7.05,1.01,3.26,1190
1,13.74,1.67,2.25,16.4,118,2.6,2.9,.21,1.62,5.85,.92,3.2,1060
1,14.22,1.7,2.3,16.3,118,3.2,3,.26,2.03,6.38,.94,3.31,970
1,13.72,1.43,2.5,16.7,108,3.4,3.67,.19,2.04,6.8,.89,2.87,1285
2,12.33,1.1,2.28,16,101,2.05,1.09,.63,.41,3.27,1.25,1.67,680
2,13.67,1.25,1.92,18,94,2.1,1.79,.32,.73,3.8,1.23,2.46,630
2,12.17,1.45,2.53,19,104,1.89,1.75,.45,1.03,2.95,1.45,2.23,355
2,13.11,1.01,1.7,15,78,2.98,3.18,.26,2.28,5.3,1.12,3.18,502
2,13.34,.94,2.36,17,110,2.53,1.3,.55,.42,3.17,1.02,1.93,750
2,12.29,1.61,2.21,20.4,103,1.1,1.02,.37,1.46,3.05,.906,1.82,870
2,13.49,1.66,2.24,24,87,1.88,1.84,.27,1.03,3.74,.98,2.78,472
2,11.96,1.09,2.3,21,101,3.38,2.14,.13,1.65,3.21,.99,3.13,886
2,13.03,.9,1.71,16,86,1.95,2.03,.24,1.46,4.6,1.19,2.48,392
2,12.33,.99,1.95,14.8,136,1.9,1.85,.35,2.76,3.4,1.06,2.31,750

2,12,.92,2,19,86,2.42,2.26,.3,1.43,2.5,1.38,3.12,278
2,12.08,1.13,2.51,24,78,2,1.58,.4,1.4,2.2,1.31,2.72,630
2,11.84,.89,2.58,18,94,2.2,2.21,.22,2.35,3.05,.79,3.08,520
2,12.16,1.61,2.31,22.8,90,1.78,1.69,.43,1.56,2.45,1.33,2.26,495
2,11.64,2.06,2.46,21.6,84,1.95,1.69,.48,1.35,2.8,1,2.75,680
2,12.08,1.83,2.32,18.5,81,1.6,1.5,.52,1.64,2.4,1.08,2.27,480
2,12.69,1.53,2.26,20.7,80,1.38,1.46,.58,1.62,3.05,.96,2.06,495
2,11.62,1.99,2.28,18,98,3.02,2.26,.17,1.35,3.25,1.16,2.96,345
2,11.81,2.12,2.74,21.5,134,1.6,.99,.14,1.56,2.5,.95,2.26,625
2,12.37,1.07,2.1,18.5,88,3.52,3.75,.24,1.95,4.5,1.04,2.77,660
2,12.08,2.08,1.7,17.5,97,2.23,2.17,.26,1.4,3.3,1.27,2.96,710
2,12.34,2.45,2.46,21,98,2.56,2.11,.34,1.31,2.8,.8,3.38,438
2,12.51,1.73,1.98,20.5,85,2.2,1.92,.32,1.48,2.94,1.04,3.57,672
2,12.25,1.73,2.12,19,80,1.65,2.03,.37,1.63,3.4,1,3.17,510
2,12.22,1.29,1.94,19,92,2.36,2.04,.39,2.08,2.7,.86,3.02,312
2,11.46,3.74,1.82,19.5,107,3.18,2.58,.24,3.58,2.9,.75,2.81,562
2,11.76,2.68,2.92,20,103,1.75,2.03,.6,1.05,3.8,1.23,2.5,607
2,12.08,1.39,2.5,22.5,84,2.56,2.29,.43,1.04,2.9,.93,3.19,385
2,11.82,1.47,1.99,20.8,86,1.98,1.6,.3,1.53,1.95,.95,3.33,495
2,12.77,3.43,1.98,16,80,1.63,1.25,.43,.83,3.4,.7,2.12,372
2,11.45,2.4,2.42,20,96,2.9,2.79,.32,1.83,3.25,.8,3.39,625
2,12.42,4.43,2.73,26.5,102,2.2,2.13,.43,1.71,2.08,.92,3.12,365
2,11.87,4.31,2.39,21,82,2.86,3.03,.21,2.91,2.8,.75,3.64,380
2,12.43,1.53,2.29,21.5,86,2.74,3.15,.39,1.77,3.94,.69,2.84,352
2,12.37,1.63,2.3,24.5,88,2.22,2.45,.4,1.9,2.12,.89,2.78,342
3,12.86,1.35,2.32,18,122,1.51,1.25,.21,.94,4.1,.76,1.29,630
3,12.81,2.31,2.4,24,98,1.15,1.09,.27,.83,5.7,.66,1.36,560
3,12.51,1.24,2.25,17.5,85,2,.58,.6,1.25,5.45,.75,1.51,650
3,12.25,4.72,2.54,21,89,1.38,.47,.53,.8,3.85,.75,1.27,720
3,13.49,3.59,2.19,19.5,88,1.62,.48,.58,.88,5.7,.81,1.82,580
3,12.93,2.81,2.7,21,96,1.54,.5,.53,.75,4.6,.77,2.31,600
3,13.52,3.17,2.72,23.5,97,1.55,.52,.5,.55,4.35,.89,2.06,520
3,12.25,3.88,2.2,18.5,112,1.38,.78,.29,1.14,8.21,.65,2,855
3,13.88,5.04,2.23,20,80,.98,.34,.4,.68,4.9,.58,1.33,415
3,13.32,3.24,2.38,21.5,92,1.93,.76,.45,1.25,8.42,.55,1.62,650

Таким образом, всего у нас получилось 93 обучающих примеров в обучающем множестве и 73 примера для тестирования работоспособности

сети в контрольном множестве. Сеть справилась с поставленной задачей всего за 15 итераций. В процессе обучения в скрытом слое нейронной сети было 7 нейронов. Точность, с которой была обучена нейронная сеть для решения поставленной задачи была определен равной $e = 0.0001$. Погрешность ошибки обучения для обучающего множества была выбрана равной $\varepsilon_1 = 0{,}001$, а для тестового $\varepsilon_2 = 0{,}01$.

Результат изменений квадратичной ошибки обучения нейронной сети изображен на следующем графике:

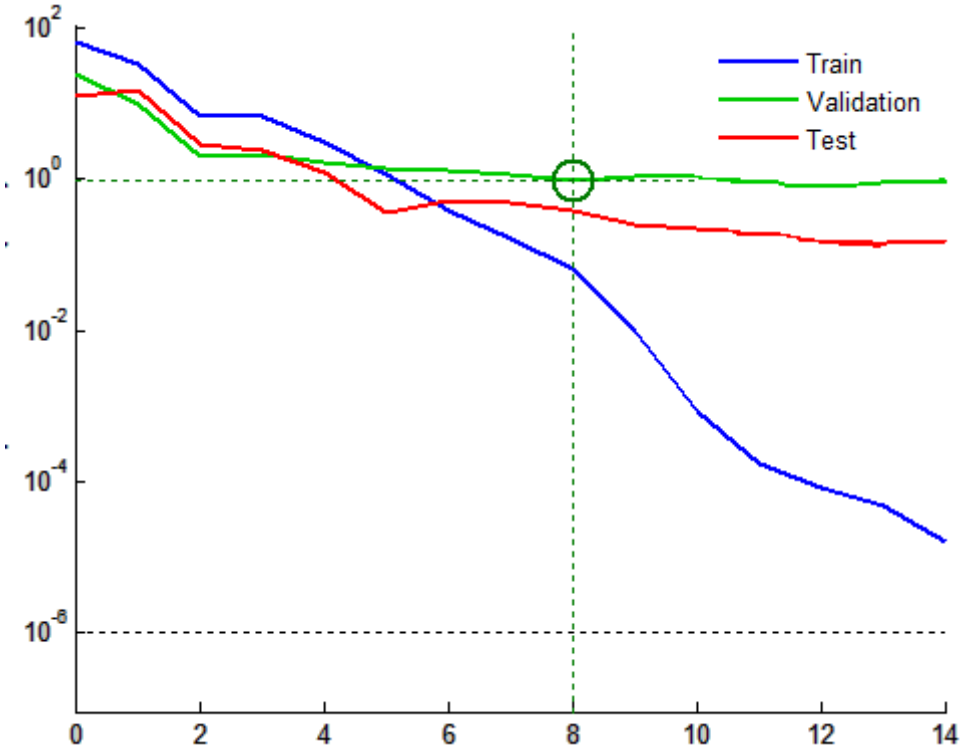

Рис. 19. Квадратичная ошибка обучения для задачи определения сортов вин.

Проведенный вычислительный эксперимент показал эффективность многослойного персептрона и сети линейной аппроксимации в решении задач аппроксимации и распознавания образов.

Заключение

На защиту выносятся следующие результаты магистерской диссертации:

1) реализован метод обучения многослойного персептрона, основанный на обратном распространении ошибки, обеспечивающий возможности борьбы с параличом обучения и избыточным обучением сети;

2) предложен пакетный метод обучения сети линейной аппроксимации и разработан алгоритм обучения;

3) реализована модификация сети линейной аппроксимации;

4) проведен вычислительный эксперимент по решению задачи аппроксимации функции двух переменных и задачи распознавания сортов вин.

Список литературы

1. Хайкин С. Нейронные сети. Полный курс. М.: Издательский дом Вильямс, 2006. 1104 с.
2. Осовский С. Нейронные сети для обработки информации. М.: Финансы и статистика, 2002. 344 с.
3. R. Rojas. Neural Networks. Springer-Verlag, 1996.
4. Bernard Widrow and Michael A. Lehr. Artificial neural networks of the perceptron and backpropagation family. Stanford University, CA 94305-4055.
5. G. Cybenko. Approximation by superpositions of sigmoidal functions. Mathematics of Control, Signals and Systems, 1989.
6. Уоссермен Ф. Нейрокомпьютерная техника. М.: Мир, 1992. 240 с.
7. Каллан Р. Основные концепции нейронных сетей. М.: Издательский дом Вильямс, 2003.- 288 с.
8. S. Gallant. Perceptron based learning algorithms. IEEE Transactions on Neural Networks, 1990.
9. Kur Horvik. Multilayer feedforward networks are universal approximators. University of California, 1989.
10. Круглов В.В., Борисов В.В. Искусственные нейронные сети. Теория и практика. М.: Горячая линия - Телеком, 2002. 382 с.
11. Прасолов В.В., Тихомиров В.М. Геометрия. - М.: МЦНМО, 2007. - 328 с.
12. Minsky M., Papert S. Perseptrons. Cambridge, VA: MIT Press, 1969.
13. Нейронные сети. Statistica Neural Networks: Пер. с англ.. М.: Горячая линия - Телеком, 2000. - 182 с.
14. Kevin Gurney, An introduction to neural networks, University of Sheffield, UCL Press, 1997.
15. Оболенский А.Ю., Оболенский И.А. Лекции по аналитической геометрии: учебно-методическое пособие. Москва-Ижевск: Институт компьютерных исследований, 2004. - 216 с.

16. Головко В.А. Нейроинтеллект: теория и применения. Книга 1. Организация и обучение нейронных сетей с прямыми и обратными связями.1999.
17. J. Minsky, S. Papert. Perceptions: An Introduction to Computational Geometry. MA. MIT Press, Cambridge, 1969.
18. В.А. Козынченко, А.И. Прус, Нейронная сеть в задачах аппроксимации, IEEE Publications , - Санкт-Петербург, 2014.